CONTENTS

ビジュアルヘルプ – FilterIIR

FilterIIR コマンドは、メニュー Analysis→Filter を選択して、ダイアログでも操作できます。
ここではダイアログと対比して説明します。

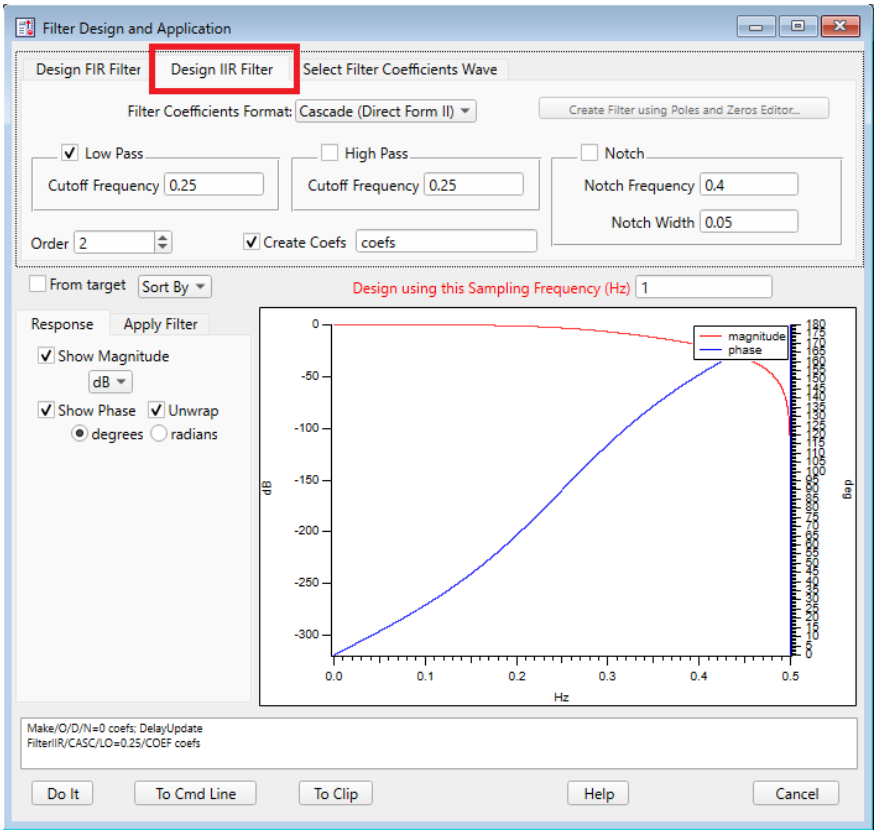
FilterIIR コマンドのヘルプ

FilterIIR [/CASC /COEF[=*coefsWaveName*] /DIM=*d* /ENDV=*ev* /HI=*fHigh* /LO=*fLow* /N={*fNotch*, *notchQ*} /ORD=*order* /Z=*z* /ZP] [*waveName*, ...]

FilterIIR コマンドでは、各 *waveName* に対して、自動的に設計された IIR フィルター係数、または *coefsWaveName* 内の IIR フィルター係数のいずれかが適用されます。
複数のフィルター設計は、複合フィルターとして統合されます。
このフィルターは、オプションで最初の *waveName* に配置することも、単に *waveName* 内のデータをフィルタリングするために使うこともできます。

自動的に設計されたフィルター係数は、Butterworth アナログプロトタイプの双線形変換であり、オプションで可変幅のノッチフィルタを備えています。

より高度な IIR フィルターを設計するには、以下の「IIR 係数の設計」のセクションを参照してください。



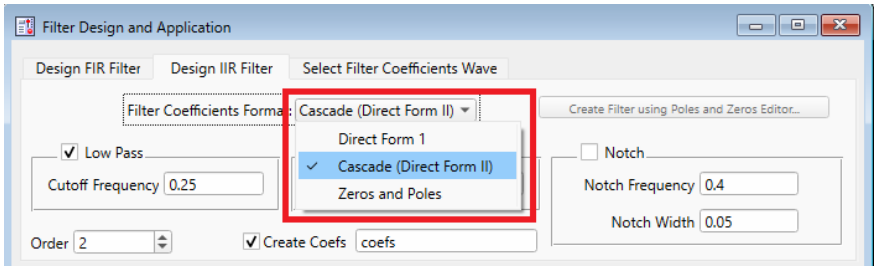
パラメーター

waveName は多次元データである場合がありますが、/DIM で指定された次元のみがフィルタリングの対象となります。
（2 次元のフィルタリングについては、MatrixFilter コマンドを参照してください。）

coefsWaveName の形式を確認する目的で、*waveName* を省略することができます。
形式に誤りがあると検出された場合、V_flag にエラーコードが返されます。
コマンドの実行が中断されないようにするには、/Z を使ってください。

フラグ

/CASC　　*coefsWaveName* にカスケード化されたバイクワッドフィルターの係数が格納されていることを指定します。高次 IIR フィルタリングにおいて、カスケード方式は Direct Form 1 フィルタリングよりも安定性が高く、数値精度も優れています。以下の「カスケードの詳細」のセクションを参照してください。



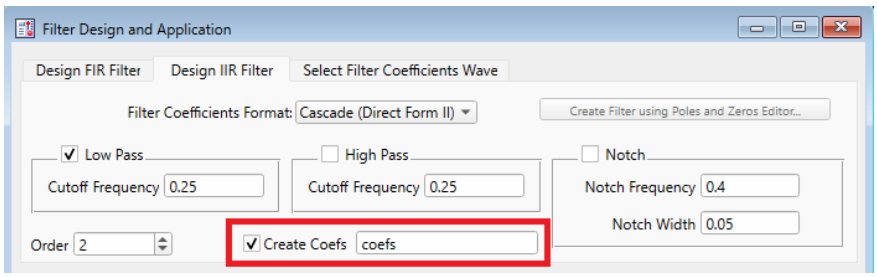
/COEF [=*coefsWaveName*]
最初の出力 *waveName* を、フィルタリングの結果ではなくフィルター係数に置き換えます。また、*coefsWaveName* が指定されている場合は、出力ウェーブを、*coefsWaveName* に含まれる IIR 係数を使って *waveName* をフィルタリングした結果に置き換えます。

coefsWaveName は、宛先の *waveNames* のいずれでもあってはなりません。また、単精度または倍精度の数値で、２次元でなければなりません。

/CASC オプションと併用する場合、*coefsWaveName* は６列で構成され、２次多項式の比の積（カスケード接続されたバイクワッドセクション）に対する実数値の係数が含まれている必要があります。

/ZP が指定された場合、その値は複素数でなければなりません。そうでない場合は、実数でなければなりません。

coefsWaveName 内の係数の形式については、「詳細」を参照してください。



/DIM=*d*　　フィルタリングするウェーブの次元を指定します。

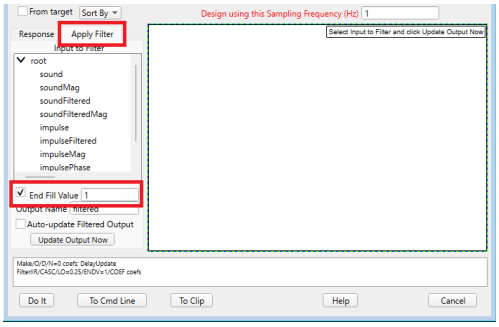
d=-1：　　ウェーブ全体を 1D として扱う（デフォルト）。

d>=0：　　行や列などに沿って処理を行います。

/DIM=0 を使うと、多次元 *waveName*（各行が特定の時点におけるすべての音声サンプルで構成される）の個々の列（それぞれがチャネル、例えば左チャネルや右チャネルなど）にフィルターを適用できます。

/ENDV=*ev*

各フィルター処理対象のウェーブの開始前の値は、*ev* に置き換えられます。/ENDV を省略した場合は、これらの値はゼロに置き換えられます。フィルター起動時のアーティファクトを防ぐため、*ev* を最初の値、またはウェーブの開始時点における局所的な平均値に設定してください。

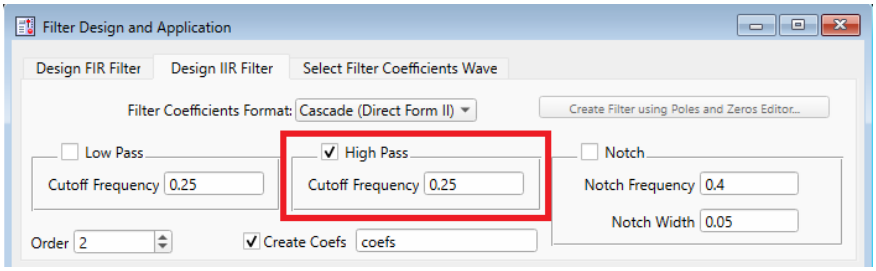


/ENDV は Igor Pro 9.0 で追加されました。

/HI=*fHigh*

fHigh を -3dB コーナー周波数とするハイパス Butterworth フィルターを作成します。フィルターの次数は /ORD フラグによってコントロールされます。

fHigh は、サンプリング周波数の分数で表されるフィルターの設計周波数で、0.5（正規化ナイキスト周波数）を超えてはなりません。

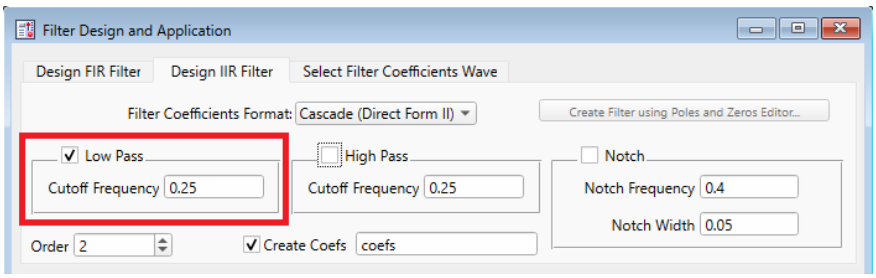


/LO=*fLow*

fLow を -3dB コーナー周波数とするローパス Butterworth フィルターを作成します。/ORD フラグはフィルターの次数をコントロールします。

fLow は、サンプリング周波数の分数で表されるフィルターの設計周波数で、0.5（正規化ナイキスト周波数）を超えてはなりません。

/HI と /LO の両方を指定して、バンドパスフィルターとバンドリジェクトフィルターを作成します。バンドパスフィルターの場合は *fLow* > *fHigh* に、バンドリジェクトフィルターの場合は *fLow* < *fHigh* に設定してください。

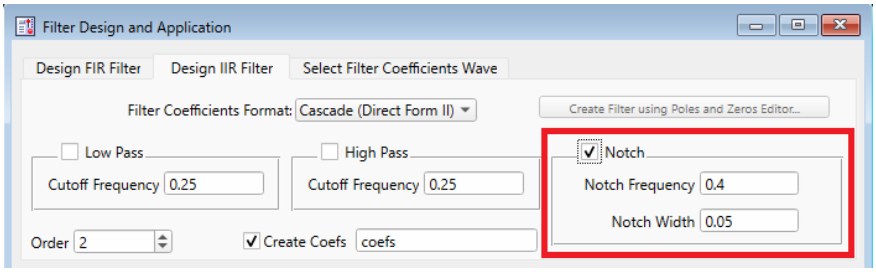


/N={*fNotch*, *notchQ*}

中心周波数が *fNotch*、-3dB 帯域幅が *fNotch* / *notchQ* のノッチフィルターを作成します。

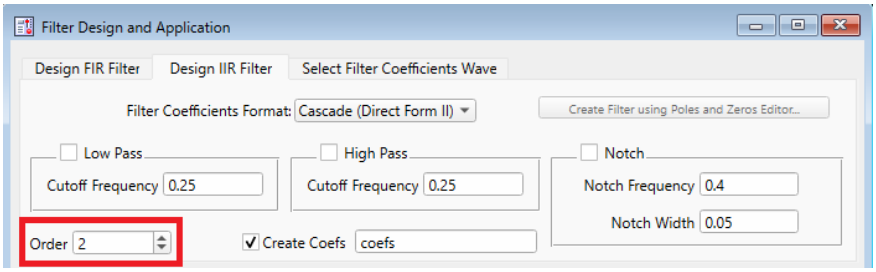
fNotch は、サンプリング周波数の分数で表されるフィルター設計周波数で、0.5（正規化ナイキスト周波数）を超えてはなりません。

notchQ は 1 より大きい数値で、通常は 10 から 100 の範囲です。この値が大きいと、フィルターの「リング」が激しくなります。



/ORD=*order*

/HI と /LO によって作成される Butterworth フィルターの次数を設定します。デフォルトは 2（2 次）で、最大値は 100 です。/HI と /LO の両方が指定された場合、ローパスフィルターとハイパスフィルターはそれぞれ order/2 の次数で作成されるため、*order* に奇数が指定された場合は、その値が次の偶数に丸められます。



/Z=*z*

エラーが発生しても、プロシージャの実行が中断されないようにします。実行を中断させるのではなく、GetRTError(1) を使ってプロシージャ内でこのケースを処理したい場合は、/Z=1 を指定してください。/Z=0 は、/Z を指定しない場合と同じです。

/ZP

coefsWaveName に、z 領域の複素零点（0 列目）および極（1 列目）が含まれていることを指定します。あるいは、*coefsWaveName* が指定されていない場合は、最初の出力 *waveName* を、zero-pole 形式のフィルター係数に置き換えることを指定します。詳細は、以下の「零点と極の詳細」のセクションを参照してください。

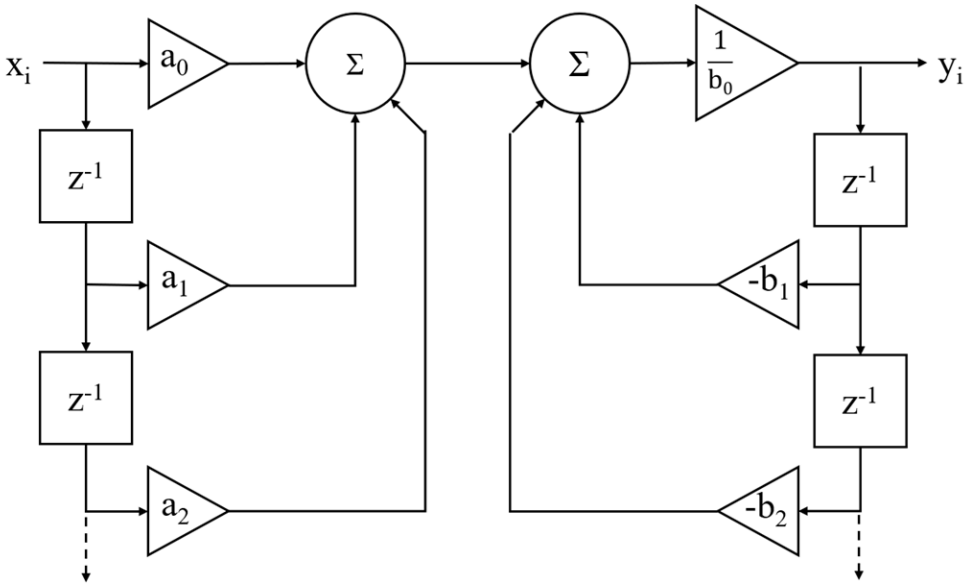
詳細

FilterIIR は、成功した場合は V_flag を 0 に設定し、エラーが発生した場合はエラーコードを設定します。
/Z フラグが設定されていない場合、エラーが発生するとコマンドの実行が停止します。
/Z を省略し、同様の状況下で GetRTError と GetRTErrMessage を呼び出すと、エラーコードの意味を確認できます。

Direct Form 1 の詳細

/CASC または /ZP が指定されていない場合、*coefsWaveName* 内の係数は、Z 変換の 2 つの多項式の比率を表します。

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots}{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots}$$



Direct Form 1 の実装

$$y_i = \frac{a_0 x_i + a_1 x_{i-1} + a_2 x_{i-2} + \cdots - b_1 y_{i-1} - b_2 y_{i-2} + \cdots}{b_0}$$

ここで、x は入力ウェーブ *waveName*、y は出力ウェーブ（*waveName* または *destWaveName* のいずれか）です。

FilterIIR は、H(z) の Direct Form I による実装を使って、フィルタリング後の結果を計算します。

coefsWaveName の 0 列目に、分母（b_i）の係数、1 列目に、分子（a_i）の係数が格納されています。

0 行目の係数は、遅延のない係数 a_0（0 列目）と b_0（1 列目）です。

1 行目の係数は、z-1 の係数である a_1 と b_1 です。

n 行目の係数は、z-n の係数である a_n と b_n です。

分子の係数の数は、分母の係数の数と異なる場合があります。
その場合は、使わない係数には 0 を指定してください。

注記

分母の係数がすべて 0 の場合（b_0 = 1 を除き、b_i = 0）、そのフィルターは実際には因果性を持つ FIR フィルター（遅延 n-1 を持つ有限インパルス応答フィルター）となります。この意味で、FilterIIR は FilterFIR コマンドのスーパーセットを実装しています。

注意

別の表記法：a_i などを分子として指定することは、Oppenheim and Shafer の『Digital Signal Processing』など多くの教科書とは異なっています。同書では、有理関数の分子係数に b、分母係数に a（暗黙的に a_0 = 1 とみなされる）を使い、残りの分母係数の符号を反転させることで、H(z) を次のように記述しています：

$$H(z)=\frac{Y(z)}{X(z)}=\frac{\sum_{i=0}^{n} b_i z^{-i}}{1-\sum_{i=1}^{n} a_i z^{-i}}$$

この表記法を使って導出された係数については、第 1 列（2 列目）の 1 行目から N 行目までに入力する前に、分母係数の符号を反転させる必要があります。また、遅延のない「欠落」分母係数である 1.0 を、0 行目、1 列目に配置する必要があります。

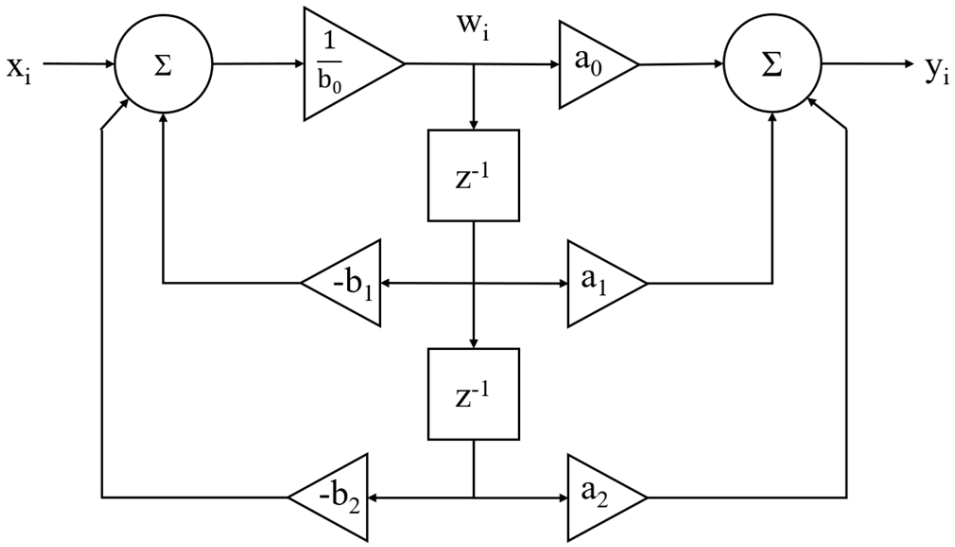
カスケードの詳細

/CASC を使う場合、*coefsWaveName* 内の係数は、Z 変換の 2 つの 2 次多項式の 1 つ以上の比の積を表します。

$$H(z)=\frac{Y(z)}{X(z)}=\prod_{k=1}^{K}\frac{a_{0_k}+a_{1_k}z^{-1}+a_{2_k}z^{-2}}{b_{0_k}+b_{1_k}z^{-1}+b_{2_k}z^{-2}}$$

各乗積項は「カスケード接続されたバイクワッドセクション」を実装しており、あるセクションの出力を次のセクションに入力することで、H(z) を実現することができます。

カスケード係数は、Direct Form II によるカスケード実装を使ってデータをフィルタリングします：



カスケード型バイクワッド Direct Form II の実装

$$w_i = \frac{x_i - b_1 w_{i-1} - b_2 w_{i-1}}{b_0}$$

$$y_i = a_0 w_i + a_1 w_{i-1} + a_2 w_{i-2}$$

高次 IIR フィルタリングにおいて、カスケード方式の実装は、Direct Form I フィルタリングよりも安定性が高く、数値精度も優れています。
フィルターの次数が 16 を超える場合は、カスケード IIR フィルタリングの使用が推奨されます（16 次の Direct Form I フィルターには、分子係数が 17 個、分母係数が 17 個あります）。

coefsWaveName は、６列からなる実数値のウェーブデータでなければなりません。
各行は１つのバイクワッドセクションを表します。
係数は次のように配置されています。

積の２番目の項（または「セクション」）（k = 2）の係数は、次の行にあり、以下同様です。

k	Row	Col 0	Col 1	Col 2	Col 3	Col 4	Col 5
1	0	a_{01}	a_{11}	a_{21}	b_{01}	b_{11}	b_{21}
2	1	a_{02}	a_{12}	a_{22}	b_{02}	b_{12}	b_{22}
…							

分子の係数（a）の数は、分母の係数（b）の数と異なっていても構いません。
この場合、使わない係数には 0 を指定してください。

例えば、３次フィルター（３極・３零点）のカスケード構成は、１次セクションと２次セクションを組み合わせたものです。
そのセクション（k）における a_{2k} と b_{2k} の値は 0 となります。
ここで、２番目のセクションは１次セクションとして指定されています。

k	Row	Col 0	Col 1	Col 2	Col 3	Col 4	Col 5
1	0	a_{01}	a_{11}	a_{21}	b_{01}	b_{11}	b_{21}
2	1	a_{02}	a_{12}	0	b_{02}	b_{12}	0
…							

注意

別の表記法：DSP に関する文献では、b_{0k} の値は通常 1 に設定され、H(z) の式には全体的なゲイン値（通常は「K」）が含まれています。
ここでは、各積項（または「セクション」）に、ユーザーが設定可能なゲイン値（b_{0k}）が割り当てられています。
整数実装においてオーバーフローを抑制するための適切なゲイン値を算出する必要があります。
浮動小数点実装の場合、全体的なゲインを制御するために、例えば b_{01} を除くすべての b_{0k} 値を 1 に設定しても構いません。

零点と極の詳細

/ZP を使う場合、*coefsWaveName* 内の係数には、（同じく複素数である）Z 変換領域における複素数の零点と極が含まれます。

$$H(z) = \frac{Y(z)}{X(z)} = \frac{(z - Z_0)(z - Z_1)(z - z_2) \ldots}{(z - p_0)(z - p_1)(z - p_2) \ldots}$$

coefsWaveName は、N+1 行からなる２列の複素数ウェーブであり、第１列には zero0、zero1、…、zeroN が、第２列には pole0、pole1、…、poleN がそれぞれ配置されている必要があります：

k	Row	Col 0	Col 1
1	0	(zero0Real, zero0Imag)	(pole0Real, pole0Imag)
2	1	(zero1Real, zero1Imag)	(pole1Real, pole1Imag)
3	2	(zero2Real, zero2Imag)	(pole2Real, pole2Imag)
…			

零点または極の虚数部が 0 以外の場合、その共役零点または共役極を *coefsWaveName* に含める必要があります。
例えば、零点が (0.7, 0.5) に配置されている場合、その共役は (0.7, -0.5) となり、その値も列 0 に記載する必要があります。
これら２つの零点は、「共役ペア」と呼ばれるものを形成します。
共役値は、1.0e-6 または 1.0e-6 * |zeroOrPole| のうち大きい方の誤差範囲内で一致している必要があります。

未使用の極や零点には (0,0) を使ってください。
z = (0,0) における零点や極は、フィルターの周波数応答に影響を与えないためです。

係数の /ZP 形式は、内部で Direct Form 1 の実装に変換されます。
また、/CASC が指定された場合は、カスケード Direct Form 2 の実装に変換されます。
これらの実装依存の係数をウェーブ形式で返すためのオプションはありません。

IIR 係数の設計

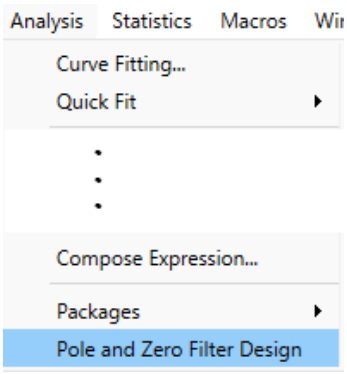
/LO、/HI、/ORD、/N、/CASC、/ZP フラグを指定することで、単純な IIR フィルターを使ったり作成したりできます。
coefsWaveName を指定せずに /COEF を使うと、これらの単純な IIR フィルターの係数が最初の *waveName* に格納されます。

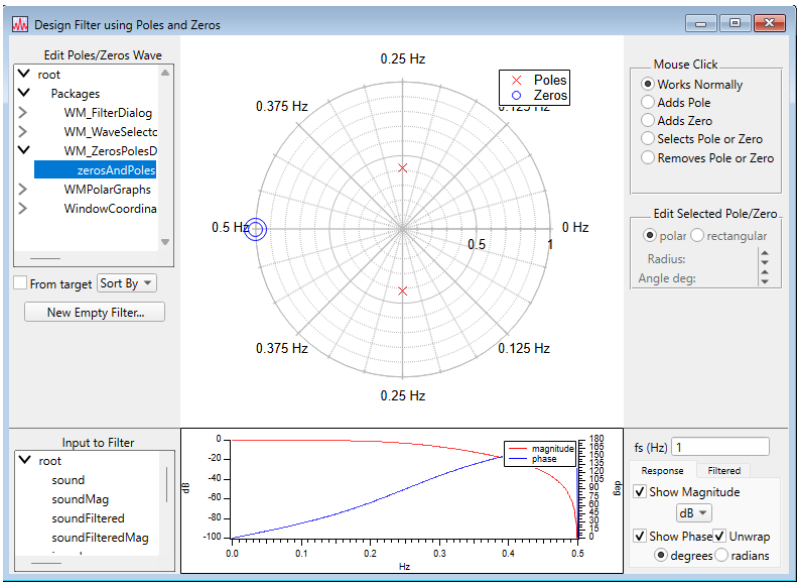
より高度な IIR フィルター（Bessel、Chebyshev）は、独立した Igor Filter Design Laboratory（IFDL）を使って設計することができます。
IFDL は、FIR（有限インパルス応答）と IIR（無限インパルス応答）フィルターを設計し、それらをデータに適用するために使う Igor Pro パッケージです。
この IIR 設計ソフトウェアは、Bessel、Butterworth、Chebyshev などのアナログプロトタイプフィルターの双線形変換に基づいて、IIR 係数を生成します。

IFDL がなくても、Pole and Zero Filter Design プロシージャを使って、Z 平面上に極と零点を手動で配置することで、カスタム IIR フィルターを作成することができます。
次の行を Procedure ウィンドウにコピーし、Procedure ウィンドウの下部にある Compile ボタンをクリックしてください。

```
#include <Pole And Zero Filter Design>
```

次に、Analysis メニューから Pole and Zero Filter Design を選択します。

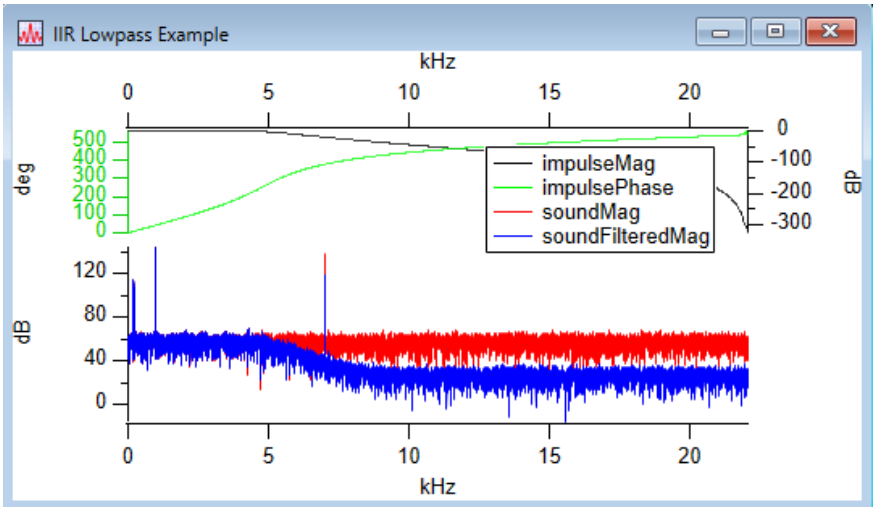




例

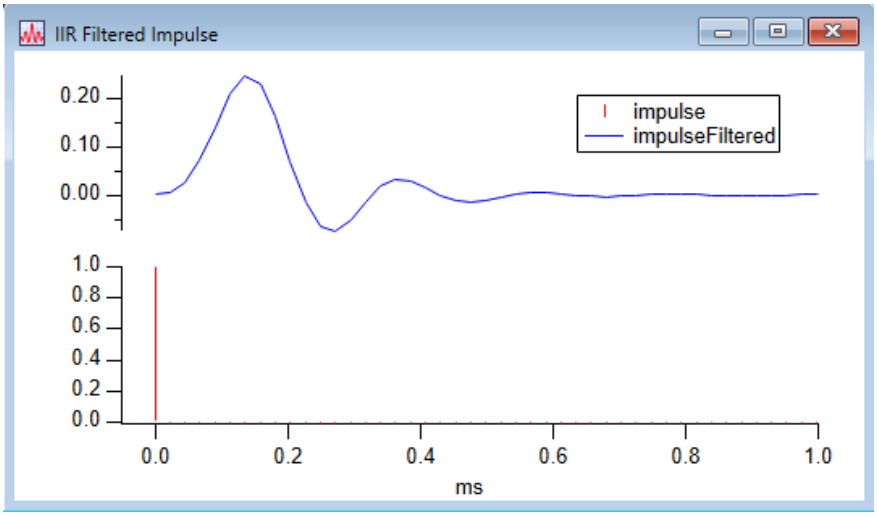
```
// ３つの正弦ウェーブからテスト音を作成する
Variable/G fs= 44100                          // サンプリング周波数
Variable/G seconds= 0.5                       // 長さ
Variable/G n= 2*round(seconds*fs/2)
Make/O/W/N=(n) sound                          // 16 ビット整数サウンドウェーブ
SetScale/p x, 0, 1/fs, "s", sound
Variable/G f1= 200, f2= 1000, f3= 7000
Variable/G a1=100, a2=3000,a3=1500
sound= a1*sin(2*pi*f1*x)
```

```
sound += a2*sin(2*pi*f2*x)
sound += a3*sin(2*pi*f3*x)+gnoise(10)          // ノイズフロアを追加

// 音のスペクトルを dB 単位で計算する
FFT/MAG/WINF=Hanning/DEST=soundMag sound
soundMag= 20*log(soundMag)
SetScale d, 0, 0, "dB", soundMag

// サウンドウェーブに 5kHz、6次ローパスフィルターを適用する
Duplicate/O sound, soundFiltered
FilterIIR/LO=(5000/fs)/ORD=6 soundFiltered     // デフォルトでは2次

// フィルタリング後の音のスペクトルを dB 単位で計算する
FFT/MAG/WINF=Hanning/DEST=soundFilteredMag soundFiltered
soundFilteredMag= 20*log(soundFilteredMag)
SetScale d, 0, 0, "dB", soundFilteredMag

// インパルスをフィルタリングして、フィルターの周波数と位相を計算する
Make/O/D/N=2048 impulse= p==0                  // t==0 でのインパルス
SetScale/P x, 0, 1/fs, "s", impulse
Duplicate/O impulse, impulseFiltered
FilterIIR/LO=(5000/fs)/ORD=6 impulseFiltered
FFT/MAG/DEST=impulseMag impulseFiltered
impulseMag= 20*log(impulseMag)
SetScale d, 0, 0, "dB", impulseMag
FFT/OUT=5/DEST=impulsePhase impulseFiltered
impulsePhase *= 180/pi                         // 度に変換
SetScale d, 0, 0, "deg", impulsePhase
Unwrap 360, impulsePhase                       // 連続相

// 周波数特性をグラフ化する
Display/R/T impulseMag as "IIR Lowpass Example"
AppendToGraph/L=phase/T impulsePhase
AppendToGraph soundMag, soundFilteredMag
ModifyGraph axisEnab(left)={0,0.6}
ModifyGraph axisEnab(right)={0.65,1}
ModifyGraph axisEnab(phase)={0.65,1}
ModifyGraph freePos=0, lblPos=60, rgb(soundFilteredMag)=(0,0,65535)
ModifyGraph rgb(impulseMag)=(0,0,0), rgb(impulsePhase)=(0,65535,0)
ModifyGraph axRGB(phase)=(3,52428,1), tlblRGB(phase)=(3,52428,1)
Legend
```




```
// フィルタリングなしとフィルタリングありのインパルス応答をグラフ化する
Display/L=leftImpulse impulse as "IIR Filtered Impulse"
AppendToGraph/L=leftFiltered impulseFiltered
ModifyGraph axisEnab(leftImpulse)={0,0.45}, axisEnab(leftFiltered)={0.55,1}
ModifyGraph freePos=0, margin(left)=50
```

```
ModifyGraph mode(impulse)=1, rgb(impulseFiltered)=(0,0,65535)
SetAxis bottom -0.00005,0.001
Legend
```




```
// 音を再生する
PlaySound sound                    // これは非常に高周波の音
PlaySound soundFiltered            // これはそうではない
```

参考文献

Embree, P.M., and B. Kimble, C Language Algorithms for Signal Processing, 456 pp., Prentice Hall, Englewood Cliffs, New Jersey, 1991.

Lynn, P.A., and W. Fuerst, Introductory Digital Signal Processing with Computer Applications, 479 pp., Prentice Hall, Englewood Cliffs, New Jersey, 1998.

Oppenheim, A.V., and R.W. Schafer, Digital Signal Processing, 585 pp., Prentice Hall, Englewood Cliffs, New Jersey, 1975.

Terrell, T.J., Introduction to Digital Filters, 2nd ed., 261 pp., John Wiley & Sons, New York, 1988.

参照

Smoothing, FilterFIR, FFT コマンド
ヘルプ IIR Filters